Doc Code: AP.PRE.REQ

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Docket Number (Optional)
50277-2250

Pursuant to 37 CFR 1.8(a)(1)(ii) I hereby certify that this correspondence is being transmitted to the United States Patent and Trademark Office via the electronic filing system in accordance with 37 CFR §§1.6(1)(4) and 1.8(a)(1)(i)(C) on the date indicated below and before 9:00 PM PST.

on ______

Signature /______

Typed or printed name ______

Application Number: 10/643,628	Filed: August 18, 2003
First Named Inventor: Wei Li	
Art Unit: 2166	Examiner: Usmaan Saeed

Applicant requests review of the final rejection in the above-identified application. No amendments are being filed with this request.

This request is being filed with a notice of appeal.
X

The review is requested for the reason(s) stated on the attached sheet(s).
Note: No more than five (5) pages may be provided.
X

I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest.
See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed. (Form PTO/SB/96)

☒ attorney or agent of record.
Registration number 57,181.

☐ attorney or agent acting under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34

/DanielDLedesma#57181/
Signature

Daniel D. Ledesma
Typed or printed name

(408) 414-1080
Telephone number

November 11, 2008
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☒ *Total of 1 forms are submitted.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Wei Li, et al.	)	Confirmation No.: 4451
	)	Examiner: Usmaan Saeed
Serial No.: 10/643,628	)	
	)	Group Art Unit No.: 2166
Filed on: August 18, 2003	)	

For: EXPRESSING FREQUENT ITEMSET COUNTING OPERATIONS

Via EFS-Web
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

PRE-APPEAL BRIEF REQUEST FOR REVIEW

Sir:

Claims 1-7, 9-20, and 22-30 are pending in the application. The Examiner made a clear factual error at least with respect to the rejection of Claim 1 under 35 U.S.C. § 103(a). Multiple features of Claim 1 are entirely absent from the cited references: U.S. Patent No. 6,324,533 issued to Agrawal et al. ("*Agrawal*") and U.S. Patent Publication No. 2002/0087561 to Chen et al. ("*Chen*"). Claim 1 recites:

> A method for performing a frequent itemset operation, the method comprising the steps of:
> within a database server that supports a particular database language, parsing a database statement to detect within the database statement, a construct that extends the particular language,
> wherein the construct identifies **a function that counts and returns frequent itemsets** given a cursor as input to the function;
> **wherein the cursor is used by the function to access values from rows that are returned from a SELECT statement;**
> **wherein the function identifies said frequent itemsets based on said values from said rows returned by said SELECT statement**;
> performing said frequent itemset operation as part of execution of the database statement to produce results; and
> storing the results in a computer-readable storage medium. (emphasis added)

At least the above-bolded features of Claim 1 are absent from *Agrawal* and *Chen*.

The Background section describes the state of the art that existed at the time the present application was filed:

> Unfortunately, there is a limit to the type of operations that SQL directly supports. Operations that are not directly supported by SQL may be performed by specifying a series of SQL operations which, when executed in combination with each other, perform the desired unsupported operation.
>
> Depending on the nature of the unsupported operation, the combination of SQL operations required to perform the unsupported operation may be quite complex. Further, amount of time and resources required to execute the series of operations may make the use of SQL impractical.
>
> An example of a type of operation that, in general, cannot be performed efficiently using SQL operations is a frequent itemset operation.
>
> When performed using available SQL operations, frequent itemset operations typically require, among other things, **so many join operations** that performance is frequently unacceptable when the operation involves any sizable item group population. (emphasis added)

Agrawal suffers the same problems identified in the Background of the Invention section. For example, FIGs. 4, 5, 8, 10, and 11 of *Agrawal* depict SQL queries that require numerous join operations and table function calls. Further, col. 6, line 61 to col. 7, line 61 teaches "a process for finding frequent itemsets." That process includes (a) candidate generation, which requires a pruning step, such as is depicted in FIG. 4, and (b) counting support to find frequent itemsets. The counting support is described in detail in col. 8 line 38 to col. 13, line 67 and depicted in FIGs. 8, 10, and 11. Counting support using SQL-92 shows the use of k-way joins, 3-way joins, subquery-based counting, and multiple group-bys. Counting support using SQL with OR extensions also teaches the use of k-way joins.

In contrast, using the invention recited in Claim 1, in order to count and return frequent itemsets, a user merely has to compose a database statement that references a <u>function</u> that:

- **counts and returns frequent itemsets;**
- **has a cursor as an input parameter;**
- **uses the cursor to access values from rows that are returned from a SELECT statement; and**
- **identifies said frequent itemsets based on said values from said rows returned by said SELECT statement.**

Agrawal and *Chen* lack any teaching or suggestion of a **function** that satisfies **any** of these limitations, much less **all** of these limitations.

1. A group-by query cannot be the recited function

The Final Office Action and the Advisory Action equate the group-by-query of *Agrarwal* with the recited function of Claim 1. This is incorrect. A group-by query, as is clear on its face, is <u>**not**</u> a function, but a <u>query</u> that includes a group-by operator. The most analogous element in Claim 1 to the group-by-query of *Agrawal* is the recited database statement. However, the group-by-query of *Agrawal* does not include a construct that identifies the recited function as Claim 1 requires.

2. The GatherComb-K table function of Agrawal *cannot be the recited function*

The Advisory Action equates the GatherComb-K table function of *Agrawal* with the recited function. This is incorrect. The GatherComb-K table function (which is just a merger of the Gather and Comb-K table functions) fails to satisfy **any** of the limitations which are explicitly required of the function recited in Claim 1:

- **counts and returns frequent itemsets;**
- **has a cursor as an input parameter;**
- **uses the cursor to access values from rows that are returned from a SELECT statement; and**
- **identifies said frequent itemsets based on said values from said rows returned by said SELECT statement.**

The Gather table function of *Agrawal* merely collects all the items of a transaction and outputs a record for the transaction (see col. 10, lines 19-22). Thus, the Gather table function neither counts nor returns frequent itemsets. Therefore, the Gather table function cannot be equated to the recited function of Claim 1.

Also, the Comb-K table function of *Agrawal* cannot be the recited function of Claim 1. According to col. 10, lines 24-27 of *Agrawal*, Comb-K takes as input the output of the Gather table function and merely ***returns*** **all k-item combinations formed out of the items of a single transaction**. For example, if k = 2 and a transaction contains items A, B, and C, then Comb-K would return {A, B}, {A, C}, and {B, C}. Although Comb-K returns one or more k-item combinations from a transaction, this is far from ***counting*** **frequent itemsets**, as Claim 1 requires. Further, there is no guarantee that any of the k-item combinations are frequent. Therefore, the Comb-K table function (as well as the GatherComb-K table function) does not even return frequent itemsets.

Furthermore, from the example database statement (in pseudo-code) in col. 10, lines 41-50 of *Agrawal*, it is clear that Gather and Comb-K are insufficient, by themselves, to count and return frequent itemsets. That example database statement shows that Gather and Comb-K are merely parts of the database statement, the **entirety** of which **must be used** to count and return frequent itemsets. Thus, the GatherComb-K table function does not **count** and **return** frequent itemsets.

3. The combination of Agrawal *and* Chen *is improper*

The Final Office Action asserts that it would have been obvious to combine *Agrawal* and *Chen* because it "would have allowed **Agrawal** to provide high concurrency for the rows in the base table when cursors are used by obtaining a lock on the rows in the base table for the duration of the cursor operation" (page 5; emphasis in Final Office Action). This is incorrect. This rationale merely refers to *Agrawal* and then copies a portion of *Chen*. The copied portion of *Chen* is in no way related to *Agrawal*'s data mining system. The goal of *Chen* is to use cursors more efficiently in order to improve concurrency for a row. However, in *Agrawal*, neither the Gather table function nor the Comb-K table function even takes a cursor as input. Thus, one of ordinary skill in the art would not even think to use the modified row-locking cursor approach of *Chen*.

In other words, in order to combine the cursors of *Chen* with *Agrawal*'s data mining system to teach or suggest the recited function of Claim 1, it **must** be alleged that it would have been obvious to use a cursor in the Gather, Comb-K, or GatherComb-K table functions of *Agrawal*. However, it would not have been obvious to use a cursor in either of these table functions. It is unclear how such a combination would even be possible, much less necessary. The Gather table function only takes two simple inputs: a transaction identifier and an item of the transaction associated with the transaction identifier. The Gather table function then outputs a record. There is no reason to modify the Gather table function to take a cursor as input. Similarly, the Comb-K table function only takes two simple inputs: a transaction identifier and a list of items of the transaction associated with the transaction identifier. There is no reason to modify the Comb-K table function to take a cursor as input.

CONCLUSION

Applicants request that the rejections of all the pending claims be reversed.